WHAT IS CLAIMED:

1. A method of modifying operation of an information system comprising:

modifying a program execution call to first execute a wrapper logic module in user space instead of a requested program;

examining a program execution request in light of one or more conditions;

selecting an action using said program execution request and said conditions;

performing said action; and

selecting a response using one or more of said program execution requests, said conditions and results of said action;

providing said selected response.

2. The method of claim 1 further comprising:

providing a wrapper logic module able to receive information about a program execution request and able to select and perform an action based on one or more conditions.

3. The method of claim 1 further wherein said conditions are one or more triggering conditions selected from the group consisting of:

program identity;

process depth;

process lineage;

external parameters;

system resource parameters;

program execution parameters;

program execution arguments;

presence or absence of other processes;

presence or absence of other programs;

presence or absence of other files;

presence or absence of other connections;

other system conditions or states;

remotely controlled settings;

availability of external decision sources;

contents, location, fingerprints, time and date stamps, and/or size of one or more of a file, a program, available or consumed memory, or input;

current, historical, and predicted states and state sequences; and

tags, cryptographic keys, and other locking and control mechanisms and their states.

4. The method of claim 1 further comprising:

setting a per-process flag indicating that a process is in a state where an program execution request should first execute said wrapper logic module;

performing error checks;

determining if said requested program would have been executed in normal operation;

executing a wrapper in place of the original program; and

providing said wrapper with relevant information regarding the original execution request.

5. The method of claim 1 further comprising:

retaining permissions associated with the original program request;

communicating with other decision processes or programs;

6. The method of claim 1 further comprising:

if a decision is made to execute the originally requested program, said wrapper replaces itself in a process space with said originally requested program; and

if a decision is made not to execute the originally requested program, said wrapper causes an alternative action to be taken.

7. The method of claim 1 further wherein said action is one or more actions selected from the group consisting of:

running a requested program;

refusing to run a requested program;

providing a response;

running a substitute program;

consulting with other local or remote programs and resources;

tunneling execution to other environments, systems, or programs through interprocess communication, networking technologies, or other communications media;

dividing execution across platforms for parallelization of tasks or to gain access to networked resources transparently to the calling process;

executing a program in altered environments or contexts such as on other computers, in 'sandbox' environments, or with altered environmental variables or simulated file system, process, memory, and I/O conditions;

modifying programs or their arguments before execution;

performing error correction and augmented or altered functions;

substituting authorized versions of programs for unauthorized versions;

associating keys to facilitate authorizations, identifications, or other augmented content or capabilities;

encrypting, decrypting, signing, or verifying programs, io, files, and other content;

sending programs to other devices or systems for encryption, decryption, signatures, or verifications;

adding or removing or modifying tags to facilitate application to tagged architectures and similar association methods of control;

running multiple versions of program and comparing results for redundancy and to assure high integrity in critical environments;

using cached results from previous executions;

changing prioritizations, locks, and scheduling to alter the normal priorities associated with program executions;

checking preconditions for program execution and automatically invoking necessary preconditions to assure proper sequencing of operations and eliminate errors and omissions;

augmenting built-in interpretation mechanisms of programs to handle more complex arguments through preprocessing;

creating or selecting suitable execution environments for otherwise uninterpretable content;

limiting available resources to a program;

consuming resources; and/or

altering privileges of the program.

8. The method of claim 1 further wherein said response comprises one or more deceptive responses that do not correspond to what was actually done by a wrapper.

9. The method of claim 1 further wherein said wrapper can provide any response it is programmed to provide, regardless of what it actually does.

10. The method of claim 1 further wherein said response is one or more responses selected from the group consisting of:

a real response of the program run;

never returning;

appearing to consume resources;

falsified legitimate responses without actually doing the requested function;

falsified responses that appear legitimate but are not;

responses that do not make sense in the context of the execution requested;

dazzlements that exhaust resources of a calling program or otherwise to cause it to fail;

responses to induce the user of a calling program to incorrectly process the resulting content;

responses that induce subsequent programs handling response data to fail to operate properly or as expected;

responses that induce syntactic or semantic errors, resonance, or dissonance in programs, people, and systems handling those results;

illegal, undefined, expected, or unexpected return values;

audit data for calibration, intrusion detection or other security systems;

set, alter, control, or interact with other deception mechanisms;

automated configuration information for calibration of detection mechanisms to particular environments;

combinations and sequences of responses consistent or inconsistent with particular environments; and

responses correlated with other response mechanisms so as to produce an overall deception in keeping with a desired deception plan or illusion.

11. The method of claim 1 further wherein a response can be combined with other responses.

12. The method of claim 1 further wherein a response can be randomly and/or selectively generated based on time, use, or other environmental or fixed factors.

13. A computer program product for use in an information system comprising:

a computer useable medium having computer readable program code embodied therein, said computer program product further comprising:

computer readable program code enabling a loadable operating system module able to intercept all program execution requests;

wherein said module, after intercepting a program execution request, initiates logic to evaluate said program execution request and determine whether to grant, refuse to grant, or falsifies granting said program execution request depending on one or more parameters; and

wherein said module, after intercepting a program execution request, returns either an accurate or an inaccurate response to said request depending on one or more parameters.

14. The computer program product of claim 13 further wherein said module further comprises:

per-process tracking logic allowing said module to ensure that program execution in a process can only be executed from a wrapper.

15. The computer program product of claim 13 further wherein:

said module can selectively return false responses in response to a program execution request.

16. The computer program product of claim 13 further wherein:

said module can probabilistically return false responses in response to a program execution request.

17. A stored program product on a media that when loaded and executed in an appropriately configured computer device enables the device to perform the method of claim 1.

18. A method of defending an information system from undesireable program execution comprising:

ensuring a per-process program execution flag is at a control state at process initiation;

checking said per-process flag at a first program execution request;

if said per-process flag is at a control state;

executing a control logic module instead of said first program;

resetting said per-process flag to an uncontrol state;

said control logic module evaluating said program execution request;

said control logic module optionally generating one or more responses;

said control logic module optionally taking one or more actions; and

said control logic module optionally issuing a different program execution request;

if said per-process flag is at an uncontrol state;

executing said program execution request; and

resetting said per-process flag to a control state.

19. The method of claim 18 further wherein said control logic module executes in a protected user space.

20. The method of claim 18 further wherein said control logic module consults other components in performing said evaluating and/or said generating.

21. A method of enhancing security in an information appliance comprising:

modifying an execution function of said information processing system to initially call a program execution evaluation module;

determining whether or not to provide deceptions; and

from said program execution evaluation module providing one or more of a set of available deceptions to entities identified for deception.

22. The method of claim 21 further wherein said program execution evaluation module executes in a protected user space.

23. A stored program product on a media that when loaded and executed in an appropriately configured computer device enables the device to perform the method of claim 21.